Exercise work 2

Name: Pekka Lehtola

Do you need help/comments:

UML Sequence diagram

Schedule:

- Week 1: Core game mechanics working.
- Week 2: UI, UML diagrams, Flow chart
- Week 3: Random events, Inheritance, Polymorphism. Ui class separated to multiple classes
- Final: Updated graphics, UML diagrams and Flow chart final version.

Challenge goal:

Challenging (Grade 4-5)

Work done this week: 12h

What has been implemented / changed:

Ui module implemented.

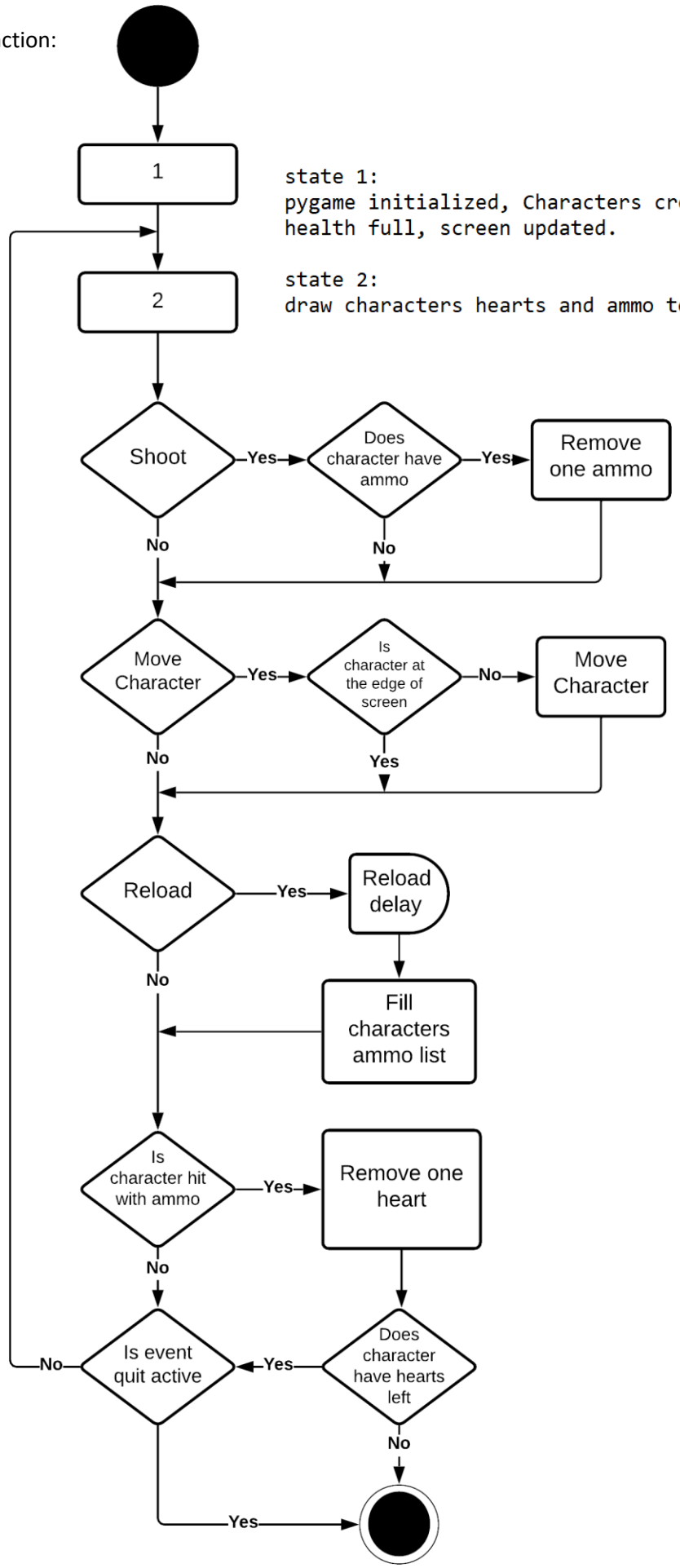
Characters shooting and reloading updated to use seconds (Previously used framerate)

Bug fixing.

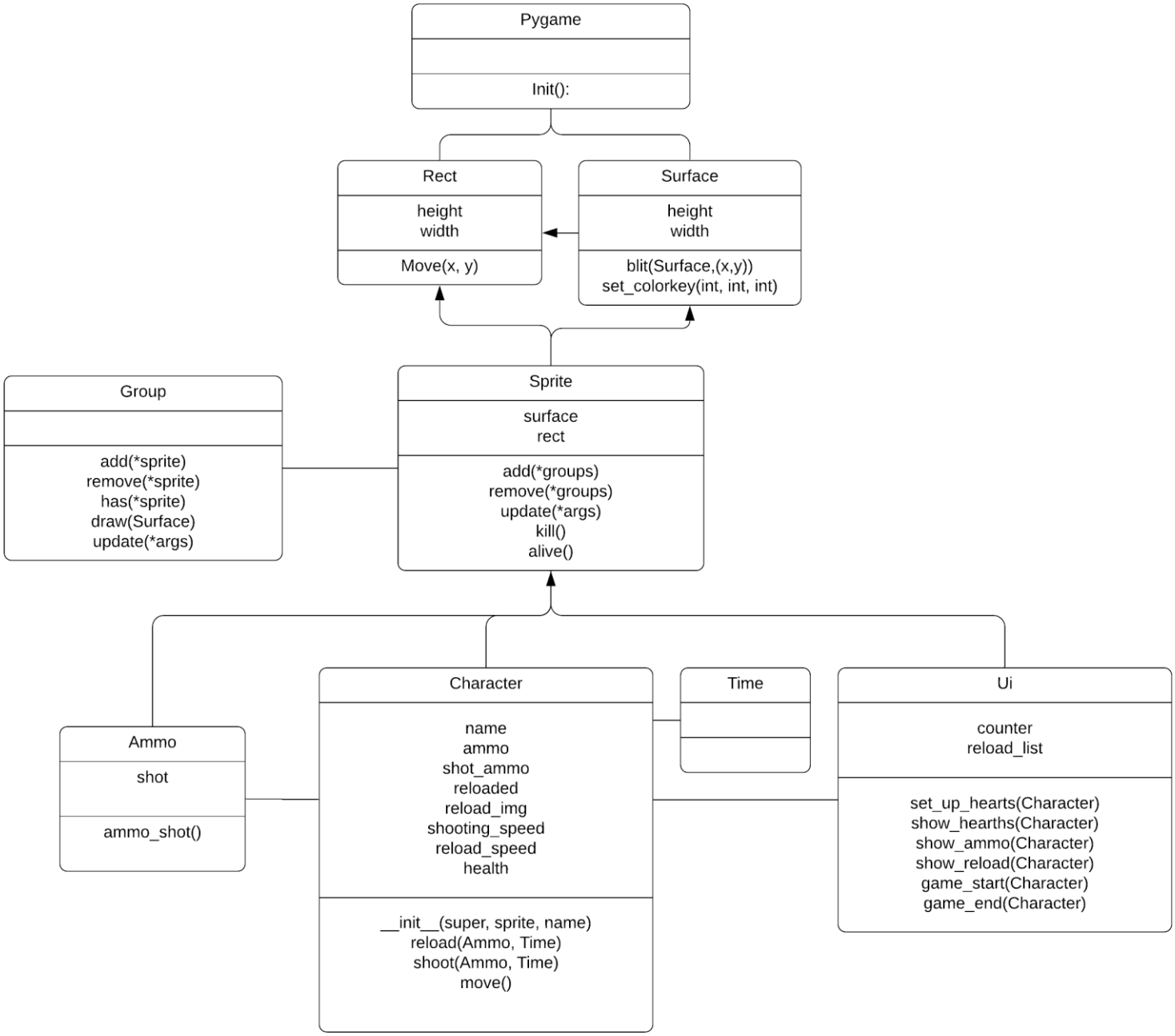
Initial flow chart and UML diagrams created.

Flow chart and UML diagrams:

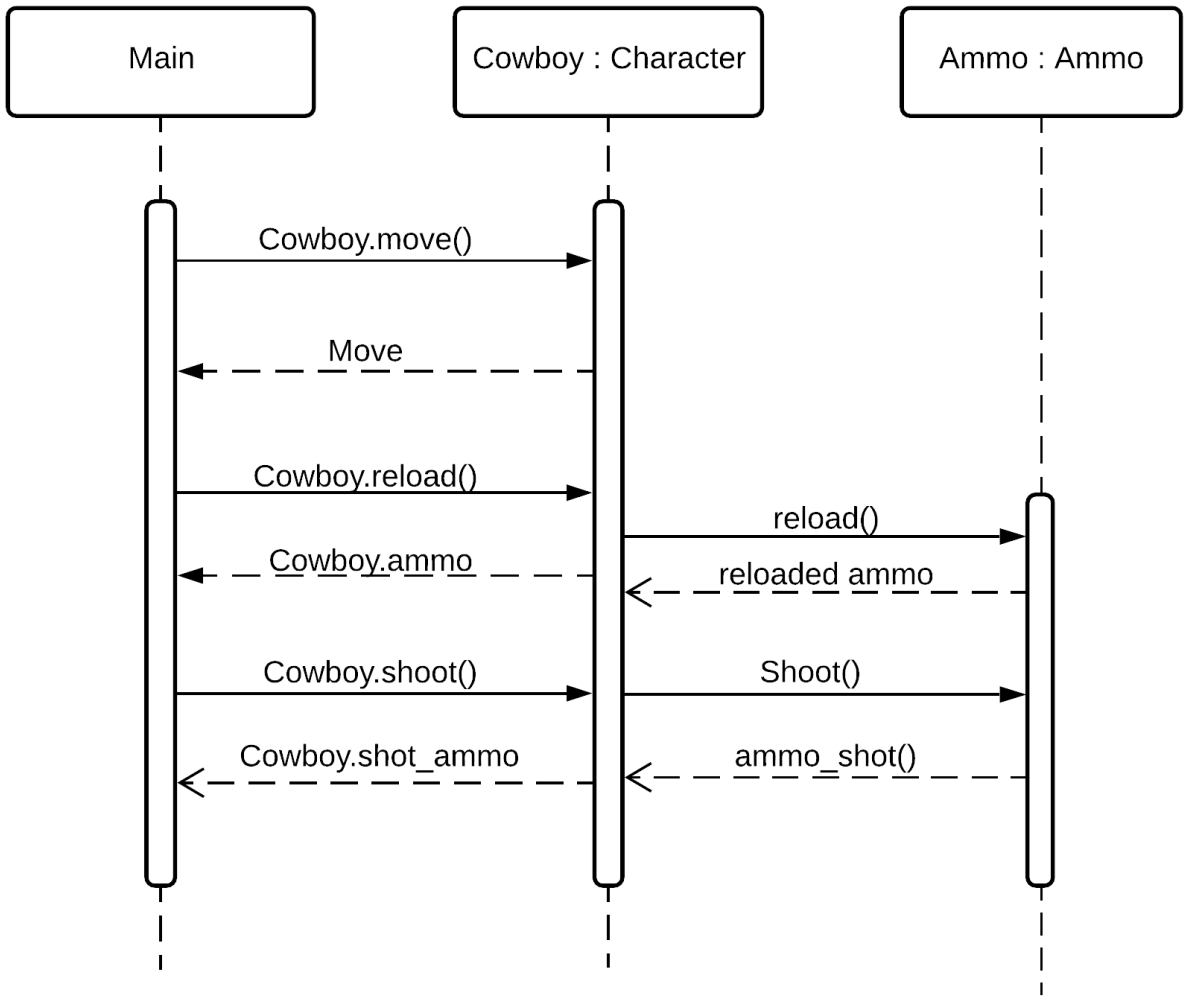
Flow chart of main function:



Class diagram:



Sequence diagram



Code:

Config:

```python
# File name: config
# Author: Pekka Lehtola
# Description: Config file for constant variables.

import pygame

FRAME_DURATION = 1/120

SCREEN_SIZE_VER = 1080
SCREEN_SIZE_HOR = 1920

BULLET_VELOCITY = 30
MOVEMENT_SPEED = 10

SHOOTING_SPEED_COWBOY = 0.4
RELOAD_SPEED_COWBOY = 2

SHOOTING_SPEED_INDIAN = 0.3
RELOAD_SPEED_INDIAN = 0.9

INF = 100**100

WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
BLACK = (0, 0, 0)

HEART = "34px"

# Green screen color is used to remove background from sprites.
GREEN_SCREEN = (0, 177, 64)

BG = pygame.image.load("images/desert.png")
ONE = pygame.image.load("images/11.png")
TWO = pygame.image.load("images/22.png")
THREE = pygame.image.load("images/33.png")
COWBOY_WINS = pygame.image.load("images/cowboy_wins.png")
INDIAN_WINS = pygame.image.load("images/indian_wins.png")

# Keyboard inputs are located here.
from pygame.locals import (
    RLEACCEL,
    K_UP,
    K_DOWN,
    K_LEFT,
    K_RIGHT,
    K_ESCAPE,
    K_w,
    K_s,
    K_d,
    K_a,
    K_SPACE,
    K_j,
    K_k,
    K_r,
    KEYDOWN,
    QUIT,
)
```

Ammo class:

```python
# File name: character
# Author: Pekka Lehtola
# Description: Ammo class for characters to shoot.

import pygame
from config import *

# Ammo is derived from pygame sprites
class Ammo(pygame.sprite.Sprite):

    def __init__(self, ammo, name):
        super(Ammo, self).__init__()
        self.name = name
        self.surf = pygame.image.load(ammo)     # Image
        self.surf.set_colorkey(GREEN_SCREEN)    # See through color
        self.rect = self.surf.get_rect()        # Hitbox

        self.shot = False

    # Function for shot ammo
    def ammo_shot(self, shooter, enemy):

        if self.shot == True:

            # If the shooter is indian arrow travels to left
            # If arrow hits cowboy, cowboy is killed.
            if shooter.name == "Indian":
                self.rect.move_ip(-BULLET_VELOCITY, 0)

                if self.rect.colliderect(enemy.rect):
                    self.rect.x = -200
                    self.kill()
                    try:
                        enemy.health.pop(-1)
                        if len(enemy.health) == 0:
                            enemy.kill()

                    except:
                        pass

            # Same as indian but bullet travels right
            if shooter.name == "Cowboy":
                self.rect.move_ip(BULLET_VELOCITY, 0)

                if self.rect.colliderect(enemy.rect):
                    self.rect.x = 2200
                    self.kill()
                    try:
                        enemy.health.pop(-1)
                        if len(enemy.health) == 0:
                            enemy.kill()
                    except:
                        pass
```

Character class 1/2:

```python
# File name: character
# Author: Pekka Lehtola
# Description: Character class that is derived from pygame sprites

import pygame
from config import *
from ammo import *

class Character(pygame.sprite.Sprite):

    def __init__(self, sprite, name):
        super(Character, self).__init__()
        self.name = name
        self.surf = pygame.image.load(sprite)                          # Surface image
        self.surf.set_colorkey(GREEN_SCREEN)                           # Defining see through color
        self.rect = self.surf.get_rect(size=(40,114), center=(0, 0))   # Sets up characters hitbox
        self.ammo = []                                                 # Reloaded ammo
        self.shot_ammo = []                                            # Ammo that character has shot
        self.reloaded = False
        self.reload_img = False
        self.shooting_speed = 0
        self.reload_speed = INF
        self.health = []

    # Reloading function

    def reload(self, pressed_keys, time):

        # Detects which player is reloading
        if self.name == "Indian":

            # Reload speed updates with timer and emptys ammo list so shooting while reload timer is on
            # is prevented
            if pressed_keys[K_k]:

                self.reload_speed = time + RELOAD_SPEED_INDIAN
                self.ammo = []
                self.reload_img = True

            # If timer is smaller than reload speed reload is set to True
            if self.reload_speed < time:
                self.reloaded = False
                self.reload_speed = INF# sets reload speed to INF to prevent automatic reloading.

            # Restarts reload speed and appends ammo to magazine.
            if self.reloaded == False:

                self.reload_img = False
                for ammo in range(0, 4):

                    ammo = Ammo("images/arrow.png", "arrow")    # Define which ammo is inserted.

                    self.ammo.append(ammo)

                self.reloaded = True


        # Cowboys reload function is indentical
        if self.name == "Cowboy":...
```

Character class 2/2 :

```
    # Shooting function
    # Work similary to reload function with time delay using time.perf_counter.
    def shoot(self, pressed_keys, time):

        if self.name == "Cowboy":

            if pressed_keys[K_SPACE] and self.shooting_speed < time:

                self.shooting_speed = time + SHOOTING_SPEED_COWBOY

                if self.reloaded:

                    try:
                        ammo = self.ammo.pop(-1)        # Removes last ammo object from ammo list
                        self.shot_ammo.append(ammo)     # Adds that ammo to shot ammo list
                        ammo.rect.x = self.rect.x + 30  # Ammos initial location is players X
                        ammo.rect.y = self.rect.y + 63  # And Y coordinates
                        ammo.shot = True

                    except:

                        pass


        # Indians shooting function is identical
        if self.name == "Indian":...


    # Moving function
    def move(self, pressed_keys):

        # Detects keyboard inputs and calculates new location for character.
        if self.name == "Indian":

            if pressed_keys[K_UP]:
                self.rect.move_ip(0, -MOVEMENT_SPEED)
            if pressed_keys[K_DOWN]:
                self.rect.move_ip(0, MOVEMENT_SPEED)
            if pressed_keys[K_LEFT]:
                self.rect.move_ip(-MOVEMENT_SPEED, 0)
            if pressed_keys[K_RIGHT]:
                self.rect.move_ip(MOVEMENT_SPEED, 0)

        if self.name == "Cowboy":

            if pressed_keys[K_w]:
                self.rect.move_ip(0, -MOVEMENT_SPEED)
            if pressed_keys[K_s]:
                self.rect.move_ip(0, MOVEMENT_SPEED)
            if pressed_keys[K_a]:
                self.rect.move_ip(-MOVEMENT_SPEED, 0)
            if pressed_keys[K_d]:
                self.rect.move_ip(MOVEMENT_SPEED, 0)

        # Detects if character has reached levels edges and stops them moving off the screen.
        if self.rect.left < 0:
            self.rect.left = 0
        if self.rect.right > SCREEN_SIZE_HOR:
            self.rect.right = SCREEN_SIZE_HOR
        if self.rect.top <= 380:
            self.rect.top = 380
        if self.rect.bottom >= SCREEN_SIZE_VER:
            self.rect.bottom = SCREEN_SIZE_VER

    ### WORK IN PROGRESS ###
    #def scale(self):
```

Ui class 1/2:

```python
# File name: ui
# Author: Pekka Lehtola
# Description: User interface class. Used for showing different elements in screen.

import pygame
from config import *
import time



class Ui(pygame.sprite.Sprite):

    def __init__(self, image):
        super(Ui, self).__init__()
        self.surf = pygame.image.load(image)
        self.surf.set_colorkey(GREEN_SCREEN)
        self.rect = self.surf.get_rect()

        #Used for reload animations.
        self.counter = 0
        self.reload_list = ["images/circle1.png", "images/circle2.png", "images/circle3.png", "images/circle4.png"]

    # Sets 4 hearts for each player.
    def set_up_hearts(self, character):

        for heart in range(0, 4):
            heart = Ui("images/hearth_1.png")
            character.health.append(heart)

    # Draws hearts for each player into the screen.
    # Heart location is determined with character name
    def show_hearts(self, character, screen):

        # height location is the same for both characters
        y = 60

        # Cowboys hearts are located in left corner
        if character.name == "Cowboy":

            x = 60

            for hearth in character.health:

                hearth.rect.y = y
                hearth.rect.x = x
                screen.blit(hearth.surf, hearth.rect)

                x += 40

        # indians hearts are located in the right corner
        if character.name == "Indian":

            x = 1826

            for hearth in character.health:

                hearth.rect.y = y
                hearth.rect.x = x
                screen.blit(hearth.surf, hearth.rect)

                x -= 40
```

Ui class 2/2:

```
    # Shows every ammo that character has in ammo list.
    def show_ammo(self, character, screen):

        y = 10

        if character.name == "Cowboy":

            x = 60

            for ammo in character.ammo:

                ammo.rect.y = y
                ammo.rect.x = x
                screen.blit(ammo.surf, ammo.rect)

                x += 30

        if character.name == "Indian":

            x = 1826

            for ammo in character.ammo:

                ammo.rect.y = y
                ammo.rect.x = x
                screen.blit(ammo.surf, ammo.rect)

                x -= 70

    # When character is reloading, shows reload animation.
    def show_reload(self, character, screen):


        if character.reload_img == True:

            if character.name == "Indian":

                self.surf = pygame.image.load(self.reload_list[int(self.counter)])
                self.surf.set_colorkey(GREEN_SCREEN)
                self.rect.y = -5
                self.rect.x = 1810
                screen.blit(self.surf, self.rect)
                self.counter += 0.2
                if self.counter >= 3.9:
                    self.counter = 0

            if character.name == "Cowboy":

                self.surf = pygame.image.load(self.reload_list[int(self.counter)])
                self.surf.set_colorkey(GREEN_SCREEN)
                self.rect.y = -5
                self.rect.x = 44
                screen.blit(self.surf, self.rect)
                self.counter += 0.2
                if self.counter >= 3.9:
                    self.counter = 0
```

Ui class work in progress:

```
    ### WORK IN PROGRESS ###

    # In the start of the game shows count down
    def game_start(self, ONE, TWO, THREE, screen):

        self.rect.x = SCREEN_SIZE_HOR / 2
        self.rect.y = SCREEN_SIZE_VER / 2

        self.surf = ONE
        self.surf.set_colorkey(GREEN_SCREEN)
        screen.blit(self.surf, self.rect)
        pygame.display.flip()
        time.sleep(1)

        self.surf = TWO
        self.surf.set_colorkey(GREEN_SCREEN)
        screen.blit(self.surf, self.rect)
        pygame.display.flip()
        time.sleep(1)

        self.surf = THREE
        self.surf.set_colorkey(GREEN_SCREEN)
        screen.blit(self.surf, self.rect)
        pygame.display.flip()
        time.sleep(1)

    # Will be used to show the winner.
    #def game_end(self):
```

Main 1/2:

```python
# File name: main
# Author: Pekka Lehtola
# Description: main file exercise work

import pygame
from config import *
from character import *
from ui import *
from time import perf_counter, sleep

running = True

def main():

        # Initialize pygame
        pygame.init()

        # Sets up screen size
        screen = pygame.display.set_mode([SCREEN_SIZE_HOR, SCREEN_SIZE_VER])

        global running

        # Creates indian and cowboy objects.
        # Also define graphigs to them.
        indian = Character("images/indian_1.png", "Indian")
        cowboy = Character("images/cowboy_1.png", "Cowboy")

        # Adds indian and cowboy to sprite group.
        all_sprites = pygame.sprite.Group()
        all_sprites.add(indian)
        all_sprites.add(cowboy)

        # Indians initial location
        indian.rect.x = 1820
        indian.rect.y = 640

        # Cowboys initial location
        cowboy.rect.x = 100
        cowboy.rect.y = 640

        hearth = Ui("images/hearth_1.png")
        reload = Ui("images/circle1.png")

        hearth.set_up_hearts(character=cowboy)
        hearth.set_up_hearts(character=indian)

        # Background update
        screen.blit(BG, (0, 0))

        # start countdown
        start = Ui("images/11.png")
        start.game_start(ONE, TWO, THREE, screen)
        start.kill()
```

Main 2/2:

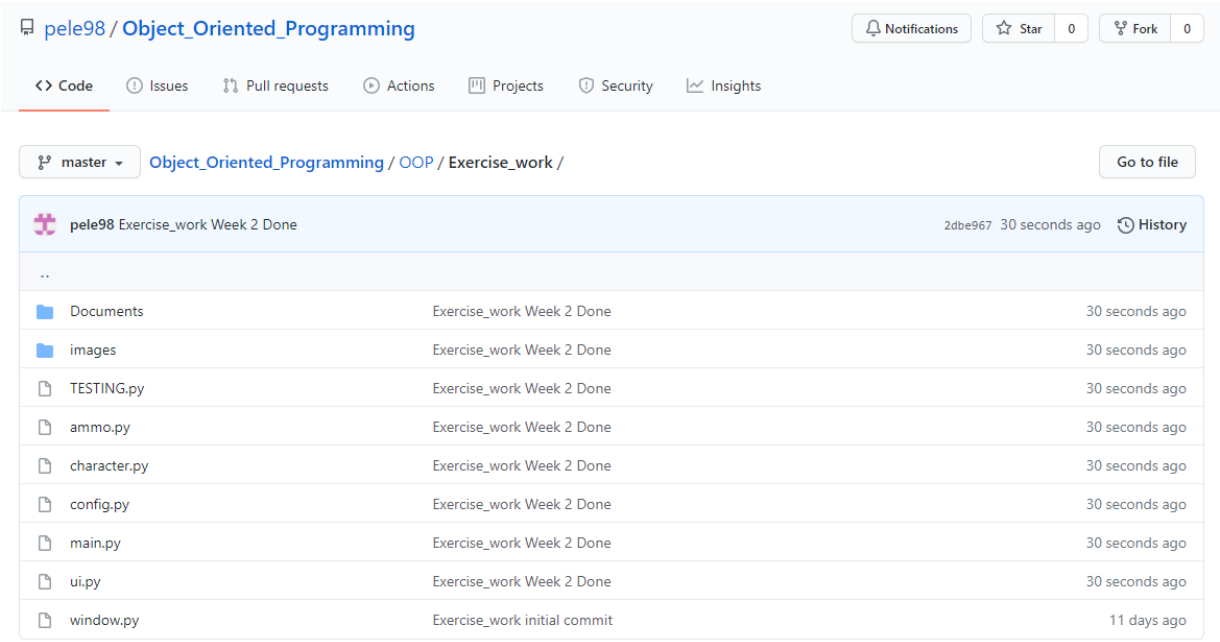
```
        while running:

            time = perf_counter()

            # Detects events while game is running.
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False

            # Pygame detects what keys are pressed.
            pressed_keys = pygame.key.get_pressed()

            # The term used for rendering objects is blitting.
            # Surf : Surface, Rect : Rectangle.
            # Rectangle is used for collide detection
            # Surface can be image or color.

            for characters in all_sprites:
                screen.blit(characters.surf, characters.rect)

            # Ui elements
            hearth.show_hearts(cowboy, screen)
            hearth.show_hearts(indian, screen)

            hearth.show_ammo(cowboy, screen)
            hearth.show_ammo(indian, screen)

            reload.show_reload(indian, screen)
            reload.show_reload(cowboy, screen)


            # Reload function
            indian.reload(pressed_keys, time)
            cowboy.reload(pressed_keys, time)

            # Shooting function
            indian.shoot(pressed_keys, time)
            cowboy.shoot(pressed_keys, time)

            # Moving function
            indian.move(pressed_keys)
            cowboy.move(pressed_keys)

            # Bullet travel and hit detection
            for ammo in indian.shot_ammo:

                ammo.ammo_shot(indian, cowboy)
                screen.blit(ammo.surf, ammo.rect)

            for ammo in cowboy.shot_ammo:

                ammo.ammo_shot(cowboy, indian)
                screen.blit(ammo.surf, ammo.rect)

            # Update screen
            pygame.display.flip()

            # Background update
            screen.blit(BG, (0,0))

            #Starts game over when either character dies.
            if len(indian.health) == 0 or len(cowboy.health) == 0:

                main()

        pygame.quit()

main()
```

Game:

Gameplay video (Sorry for the quality…):

https://youtu.be/icjRZ-IOyUQ

Screen capture of git log (showing that you made a commit after every task).



Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Ok. Kuvien animointi toi tiettyjä haasteita ja UML diagrammien teko oli haastavaa. Muuten ohjelmointi sujui kuin rasvattu.

Doing this exercise, I learned…

Miten tehdä animaatioita.

I am still wondering…

-

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

Sequence diagrammi omasta mielestä ei ole tarpeeksi hyvä.